Technical Report

TR-2014-007

Updating and Downdating Techniques for Optimizing Network Communicability

by

Francesca Arrigo, Michele Benzi

Mathematics and Computer Science

EMORY UNIVERSITY

UPDATING AND DOWNDATING TECHNIQUES FOR OPTIMIZING NETWORK COMMUNICABILITY

FRANCESCA ARRIGO[†] AND MICHELE BENZI[‡]

Abstract. The total communicability of a network (or graph) is defined as the sum of the entries in the exponential of the adjacency matrix of the network, possibly normalized by the number of nodes. This quantity offers a good measure of how easily information spreads across the network, and can be useful in the design of networks having certain desirable properties. The total communicability can be computed quickly even for large networks using techniques based on the Lanczos algorithm.

In this work we introduce some techniques that can be used to construct undirected networks that are highly sparse, and yet have a large total communicability. These methods are based on updating, downdating and rewiring techniques that take into account the change in total communicability resulting from the addition or deletion of an edge. To this end, we introduce new edge centrality measures which can be used to guide in the selection of edges to be added or removed.

Moreover, we show experimentally that the total communicability provides an effective and easily computable measure of how "well-connected" a sparse network is.

Key words. network analysis; eigenvector centrality; subgraph centrality; total communicability; edge centrality; natural connectivity.

AMS subject classifications. 05C50, 15A16, 65F60

1. Introduction.

Network models are nowadays ubiquitous in the natural, information, social, and engineering sciences. The last 15 years or so have seen the emergence of the vast, multidisciplinary field of Network Science, with contributions from a wide array of researchers including physicists, mathematicians, computer scientists, engineers, biologists, and social scientists [2, 16, 34]. Applications of Network Science range from biology to public health, from social network analysis to homeland security, from economics to the humanities, from marketing to information retrieval. Network analysis is also an essential ingredient in the design of information, communication, and transportation networks, as well as in energy-related disciplines such as power grid maintenance, control, and optimization [36]. Graph theory and linear algebra provide abstractions and quantitative tools that can be employed in the analysis and design of large and complex network models.

Real-world networks are characterized by structural properties that make them very different from both regular graphs on one hand, and completely random graphs on the other. Real networks frequently exhibit a highly skewed degree distribution (often following a power law), small diameter, high clustering coefficient (the two last properties together are often referred to as the *small world* property), the presence of motifs, communities, and other signatures of complexity.

Some of the basic questions in network analysis concern node and edge centrality, community detection, communicability, and diffusion [10, 16, 34]. Related to these are the important notions of network robustness (or its opposite, vulnerability) and connectivity [12]. These latter properties refer to the degree of resiliency displayed by the network in the face of random accidental failures or deliberate, targeted attacks, which can be modeled in terms of edge or node removal. Generally speaking, it is

[†]Department of Science and High Technology, University of Insubria, Como 22100, Italy (francesca.arrigo@uninsubria.it).

[‡]Department of Mathematics and Computer science, Emory University, Atlanta, Georgia 30322, USA (benzi@mathcs.emory.edu). The work of this author was supported by National Science Foundation grants DMS1115692 and DMS-1418889.

desirable to design networks that are at the same time highly sparse (in order to reduce costs) and highly connected, meaning that disconnecting or disrupting the network would require the removal of a large number of edges. Such networks should not contain bottlenecks, and they should allow for the rapid exchange of communication between nodes. Expander graphs [15, 26] are an important class of graphs with such properties.

Moreover, there are situations (such as counterterrorism operations or the containment of infectious deseases) where one may actually want to reduce as much as possible communication between nodes while modifying the network as little as possible. Clearly, this problem is closely connected to the previous one.

In this paper we describe some techniques that can be brought to bear on the problems described above and related questions. Our approach is based on the notion of *total communicability* of a network, introduced in [6] on the basis of earlier work by Estrada and Hatano [17, 18]. Total communicability, defined as the (normalized) sum of the entries in the exponential of the adjacency matrix of the network, provides a global measure of how well the nodes in a graph can exchange information. Communicability is based on the number and length of graph walks connecting pairs of nodes in the network. Pairs of nodes (i, j) with high communicability correspond to large entries $[e^A]_{ij}$ in the matrix exponential of A, the adjacency matrix of the network.

Total network communicability can also be used to measure the connectivity of the network as a whole. For instance, given two alternative network designs (with a similar "budget" in terms of number of candidate edges), one can compare the two designs by computing the respective total communicabilities and pick the network with the highest one, assuming that a well-connected network with high node communicability is the desired goal. It is important to stress that the total communicability of a network can be efficiently computed or estimated even for large networks using Lanczos or Arnoldi based algorithms without having to compute any individual entry of e^A (only the ability to perform matrix-vector products with A is required).

In this paper we consider the following problems:

- Given an existing connected (undirected) network $G = (V, E)$, choose an existing edge $(i, j) \in E$ such that removing this edge would minimize the decrease in the total communicability of the network (and preserve the network connectedness); this is repeated until a prescribed reduction in cost (measured in terms of deleted edges) has been achieved. We refer to this as the *downdating problem*.
- Given an existing sparse (undirected) network $G = (V, E)$, choose a pair of nodes $i, j \in V$ (with $i \neq j$ and $(i, j) \notin E$) such that adding the edge (i, j) to E would maximize the increase in the total communicability of the network; this is then repeated until a prescribed budget of new edges has been exhausted. We refer to this as the *updating problem*.
- Given an existing sparse connected (undirected) network $G = (V, E)$, perform a sequence of downdate-then-update steps, with the goal of maximizing the total communicability of the modified network; we call this the *rewiring problem*.

The importance of the first two problems for network analysis and design is obvious. We note that an efficient solution to the second problem would also suggest how to proceed if the goal is to identify existing edges whose removal would *maximize* the decrease in communicability, which could be useful, e.g., in planning anti-terrorism operations or public health policies. The third problem is motivated by the obser-

vation that for transport networks (e.g., flight routes) it is sometimes desirable to redirect edges in order to improve the performance (i.e., increase the number of travellers) without increasing too much the costs. Hence, in such cases, one wants to eliminate a route used only by a few travellers and to add a route that may be used by a lot of people.

The above problems may arise not only in the design of infrastructural networks (such as telecommunication or transportation networks), but also in other contexts. For instance, in social networks the addition of a friendship/collaborative tie may change dramatically the structure of the network, leading to a more cohesive group, and hence preventing the splitting of the community into smaller subgroups.

In this paper we develop updating, downdating, and rewiring techniques which manipulate the edges of the network in such a way that the total communicability responds as desired to these modifications.

The work is organized as follows. Section 2 contains some basic facts from linear algebra and graph theory, and introduces the modifications of the adjacency matrix we will perform. In section 3 we derive bounds for the total communicability via the Gauss–Radau quadrature rule and we describe how these bounds change when a rank-two modification of the adjacency matrix is performed. Section 4 is devoted to the introduction of the methods to controllably modify the graph in order to adjust the value of its total communicability, together with the corresponding complexity estimates. Numerical studies to assess the effectiveness and performance of the techniques introduced are provided in section 5 for both synthetic and real-world networks. In section 6 we discuss the evolution of a popular measure of network connectivity, known as the natural connectivity, when the same modifications are performed. This section provides further evidence that motivates the use of the total communicability as a measure of connectivity. Finally, in section 7 we draw conclusions and we describe future directions.

2. Background and definitions. In this section we provide some basic definitions, notations, and properties associated with graphs.

A *graph* or *network* $G = (V, E)$ is defined by a set of n nodes (vertices) V and a set of m edges $E = \{(i, j)|i, j \in V\}$ between the nodes. An edge is said to be *incident* to a vertex i if there exists a node $j \neq i$ such that either $(i, j) \in E$ or $(j, i) \in E$. The *degree* of a vertex, denoted by d_i, is the number of edges incident to i in G. A graph is *regular* if all nodes have the same degree. The graph is said to be *undirected* if the edges are formed by unordered pairs of vertices. A *walk* of length k in G is a set of nodes $i_1, i_2, \ldots, i_k, i_{k+1}$ such that for all $1 \leq l \leq k$, $(i_l, i_{l+1}) \in E$. A *closed walk* is a walk for which $i_1 = i_{k+1}$. A *path* is a walk with no repeated nodes. A graph is *connected* if there is a path connecting every pair of nodes. A graph with unweighted edges, no self-loops (edges from a node to itself), and no multiple edges is said to be *simple*. Throughout this work, we will consider undirected, simple, and connected networks.

Every graph can be represented as a matrix $A = (a_{ij}) \in \mathbb{R}^{n\times n}$, called the *adjacency matrix* of the graph. The entries of the adjacency matrix of an unweighted graph $G = (V, E)$ are

$$a_{ij} = \begin{cases} 1 & \text{if } (i, j) \in E \\ 0 & \text{otherwise} \end{cases} \qquad \forall i, j \in V.$$

If the network is simple, the diagonal elements of the adjacency matrix are all equal to zero. In the special case of an undirected network, the associated adjacency matrix

is symmetric, and thus its eigenvalues are real.

We label the eigenvalues in non-increasing order: $\lambda_1 \geq \lambda_2 \geq \ldots \geq \lambda_n$. Since A is a real-valued, symmetric matrix, we can decompose A into $A = Q\Lambda Q^T$ where Λ is a diagonal matrix containing the eigenvalues of A and $Q = [\mathbf{q}_1, \ldots, \mathbf{q}_n]$ is orthogonal, where $\mathbf{q}_i$ is an eigenvector associated with λ_i. Moreover, if G is connected, A is irreducible and from the Perron–Frobenius Theorem [32, Chapter 8] we deduce that $\lambda_1 > \lambda_2$ and that the leading eigenvector $\mathbf{q}_1$, sometimes referred to as the *Perron vector*, can be chosen such that its components $q_1(i)$ are positive for all $i \in V$.

We can now introduce the basic operations which will be performed on the adjacency matrix A associated with the network $G = (V, E)$. We define the *downdating* of the edge $(i, j) \in E$ as the removal of this edge from the network. The resulting graph $\widehat{G} = (V, \widehat{E})$, which may be disconnected, has adjacency matrix

$$\widehat{A} = A - UW^T, \qquad U = [\mathbf{e}_i, \mathbf{e}_j], \quad W = [\mathbf{e}_j, \mathbf{e}_i],$$

where here and in the rest of this work the vectors $\mathbf{e}_i$, $\mathbf{e}_j$ represent the ith and jth vectors of the standard basis of $\mathbb{R}^n$, respectively.

Similarly, let $(i, j) \in \overline{E}$ be an element in the complement of E. We will call this element a *virtual edge* for the graph G. We can construct a new graph $\tilde{G} = (V, \tilde{E})$ obtained from G by adding the virtual edge (i, j) to the graph. This procedure will be referred to as the *updating* of the virtual edge (i, j). The adjacency matrix of the resulting graph is

$$\tilde{A} = A + UW^T, \qquad U = [\mathbf{e}_i, \mathbf{e}_j], \quad W = [\mathbf{e}_j, \mathbf{e}_i].$$

Hence, these two operations can both be described as rank-two modifications of the adjacency matrix of the original graph.

The operation of downdating an edge and successively updating a virtual edge will be referred to as *rewiring*.

REMARK 1. These operations are all performed in a symmetric fashion, since in this paper we consider exclusively undirected networks.

2.1. Centrality and total communicability. One of the main goals when analyzing a network is to identify the most influential nodes in the network. Over the years, various measures of the importance, or centrality, of nodes have been developed [10, 16, 34]. In particular the *(exponential) subgraph centrality* of a node i (see [19]) is defined as the ith diagonal element of the matrix exponential [25]:

$$e^A = I + A + \frac{A^2}{2!} + \ldots = \sum_{k=0}^{\infty} \frac{A^k}{k!},$$

where I is the $n \times n$ identity matrix. As it is well known in graph theory, given an adjacency matrix A of an unweighted network and $k \in \mathbb{N}$, the element $\left(A^k\right)_{ij}$ counts the total number of walks of length k starting from node i and ending at node j. Therefore, the subgraph centrality of node i counts the total number of closed walks centered at node i, weighting walks of length k by a factor $\frac{1}{k!}$, hence giving more importance to shorter walks. The subgraph centrality then accounts for the returnability of the information to the node which was the source of this same information. Likewise, the off-diagonal entries of the adjacency matrix $\left(e^A\right)_{ij}$ (*subgraph communicability* of nodes i and j) account for the ability of nodes i and j of exchanging information [17, 18].

Starting from these observations and with the aim of reducing the cost of the computation of the rankings, in [6] it was suggested to use as a centrality measure the *total communicability of a node* i, defined as the ith entry of the vector $e^A\mathbf{1}$, where $\mathbf{1}$ denotes the vector of all ones. This measure of centrality for a node counts both its ability of spreading information along the network and the returnability of the information to the node itself.

The value resulting from summing these quantities over all the nodes can be interpreted as a global measure of how effectively the communication takes place across the whole network. This index is called *total (network) communicability* [6] and can be written as

$$TC(A) := \mathbf{1}^T e^A \mathbf{1} = \sum_{i=1}^{n}\sum_{j=1}^{n}(e^A)_{ij} = \sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{k=1}^{n} e^{\lambda_k} q_k(i) q_k(j). \tag{2.1}$$

This value can be efficiently computed, e.g., by means of a Krylov method as implemented in S. Güttel's Matlab toolbox `funm_kryl` see [1, 23] or by Lanczos-based techniques as discussed below. In the toolbox [23] an efficient algorithm for evaluating $f(A)\mathbf{v}$ is implemented; with this method the vector $e^A\mathbf{1}$ can be constructed in roughly $O(n)$ operations (note that the prefactor can vary for different types of networks) and the total communicability is easily derived.

As it is clear from its definition, the value of $TC(A)$ may be very large. Several normalizations have been proposed; the simplest is the normalization by the number of nodes n (see [6]), which we will use throughout the paper.

3. Bounds via quadrature rules. For an undirected network it is easy to prove that the normalized total communicability satisfies

$$\frac{1}{n}\sum_{i=1}^{n}\left(e^A\right)_{ii} \le \frac{TC(A)}{n} \le e^{\lambda_1}, \tag{3.1}$$

where the lower bound is attained by the graph with n nodes and no edges and the upper bound is attained by the complete graph with n nodes.

More refined bounds for this index can be obtained by means of quadrature rules, as described in [4, 5, 22]. Indeed, bounds on bilinear forms $\mathbf{u}^T f(A)\mathbf{v}$ can be derived based on Gauss–type quadrature rules when f is a *strictly completely monotonic* (s.c.m.) function on the interval $[a, b]$ containing the spectrum of A by working on a 2×2 matrix derived from one step of the symmetric Lanczos iteration (see [5, 22]). Recall that a function is s.c.m. on $[a, b]$ if $f^{(2l)}(x) > 0$ and $f^{(2l+1)}(x) < 0$ for all $x \in [a, b]$ and for all $l \ge 0$, where $f^{(k)}$ denotes the kth derivative of f and $f^{(0)} \equiv f$. In this particular context, this means that we need to use $f(x) = e^{-x}$ and therefore to work on the matrix $-A$ in order to compute bounds for the normalized total communicability.

The starting point is to observe that bilinear forms $\mathbf{u}^T f(A)\mathbf{v}$ can be thought of as Riemann–Stieltjes integrals with respect to the spectral measure associated with the symmetric matrix A:

$$\mathbf{u}^T f(A)\mathbf{v} = \int_a^b f(\lambda)dm(\lambda), \quad m(\lambda) = \begin{cases} 0, & \lambda < a = \lambda_n \\ \sum_{k=i+1}^{n} w_k z_k, & \lambda_{i+1} \le \lambda < \lambda_i \\ \sum_{k=1}^{n} w_k z_k, & b = \lambda_1 \le \lambda \end{cases}$$

where $A = Q\Lambda Q^T$, $\mathbf{w} = Q^T\mathbf{u} = (w_i)$, and $\mathbf{z} = Q^T\mathbf{v} = (z_i)$.

This integral can be approximated by means of Gauss–type quadrature rules, which can be used to obtain lower and upper bounds on the bilinear forms of interest. In particular, our bounds are derived using the Gauss–Radau quadrature rule:

$$\int_a^b f(\lambda)dm(\lambda) = \sum_{j=1}^{p} c_j f(t_j) + v_1 f(\tau_1), \tag{3.2}$$

where the nodes $\{t_j\}_{j=1}^p$ and the weights $\left\{\{c_j\}_{j=1}^p, v_1\right\}$ are to be determined, whereas τ_1 is prescribed and equal either to a or to b. The Gauss–Radau bounds are then as described in the following theorem.

THEOREM 3.1 (6.4 in [22]). *Suppose f is such that $f^{(2l+1)}(\xi) < 0$ for all l and for all $\xi \in (a,b)$. Let U_{GR} be defined as*

$$U_{GR}[f] = \sum_{j=1}^{p} c_j^a f(t_j^a) + v_1^a f(a),$$

c_j^a, v_1^a, t_j^a being the weights and nodes in (3.2) with $\tau_1 = a$, and let L_{GR} be defined as

$$L_{GR}[f] = \sum_{j=1}^{p} c_j^b f(t_j^b) + v_1^b f(b),$$

c_j^b, v_1^b, t_j^b being the weights and nodes in (3.2) with $\tau_1 = b$. The Gauss–Radau rule is exact for polynomials of degree less than or equal to $2p$ and satisfies

$$L_{GR}[f] \leq \int_a^b f(\lambda)dm(\lambda) \leq U_{GR}[f].$$

Moreover for all p there exists $\eta_U, \eta_L \in [a,b]$ such that

$$\int_a^b f(\lambda)dm(\lambda) - U_{GR}[f] = \frac{f^{(2p+1)}(\eta_U)}{(2p+1)!} \int_a^b (\lambda - a) \left[\prod_{j=1}^{p} (\lambda - t_j^a)\right]^2 dm(\lambda),$$

$$\int_a^b f(\lambda)dm(\lambda) - L_{GR}[f] = \frac{f^{(2p+1)}(\eta_L)}{(2p+1)!} \int_a^b (\lambda - b) \left[\prod_{j=1}^{p} (\lambda - t_j^b)\right]^2 dm(\lambda).$$

It is therefore necessary to evaluate two quadrature rules, one for the upper bound and one for the lower bound. However, the explicit computation of nodes and weights can be avoided. Indeed, the evaluation of the quadrature rules is mathematically equivalent to the computation of orthogonal polynomials via a three–term recurrence, or, equivalently, to the computation of entries and spectral information of a certain tridiagonal matrix via the Lanczos algorithm. In fact, the right hand side of equation (3.2) can be computed from the relation (Theorem 6.6 in [22]):

$$\sum_{j=1}^{p} c_j f(t_j) + v_1 f(\tau_1) = \mathbf{e}_1^T f(J_{p+1}) \mathbf{e}_1, \tag{3.3}$$

where

$$J_{p+1} = \begin{pmatrix} \omega_1 & \gamma_1 & & & \\ \gamma_1 & \omega_2 & \gamma_1 & & \\ & \ddots & \ddots & \ddots & \\ & & \gamma_{p-1} & \omega_p & \gamma_p \\ & & & \gamma_p & \omega_{p+1} \end{pmatrix}$$

is a tridiagonal matrix whose eigenvalues are the Gauss–Radau nodes (and hence J_{p+1} is built so as to have the prescribed eigenvalue τ_1), whereas the weights are given by the squares of the first entry of the normalized eigenvectors of J_{p+1}. An efficient implementation of this technique is provided in G. Meurant's `mmq` toolbox for Matlab [31]. This toolbox, adapted to handle sparsity, will be used for some of the numerical experiments presented later in the paper.

We can now state our result on the bounds for the normalized total network communicability.

THEOREM 3.2. *Let A be the adjacency matrix of an unweighted and undirected network. Then*

$$\Phi\left(\beta, \omega_1 + \frac{\gamma_1^2}{\omega_1 - \beta}\right) \le \frac{TC(A)}{n} \le \Phi\left(\alpha, \omega_1 + \frac{\gamma_1^2}{\omega_1 - \alpha}\right)$$

where $[\alpha, \beta]$ is an interval containing the spectrum of $-A$ (i.e., $\alpha \le -\lambda_1$ and $\beta \ge -\lambda_n$), $\omega_1 = -\mu = -\frac{1}{n}\sum_{i=1}^n d_i$ is the negative mean of the degrees, $\gamma_1 = \sigma = \sqrt{\frac{1}{n}\sum_{k=1}^n (d_k - \mu)^2}$ is the standard deviation, and

$$\Phi(x, y) = \frac{c\,(e^{-x} - e^{-y}) + xe^{-y} - ye^{-x}}{x - y}, \qquad c = \omega_1. \tag{3.4}$$

Proof. First we derive an explicit expression for the right–hand side of equation (3.3) when $f(x) = e^{-x}$ and J_2 is 2×2 with the help of the Lagrange interpolation formula for the evaluation of matrix functions [25]. Let μ_1 and μ_2 be distinct eigenvalues of a given 2×2 matrix $B = (b_{ij})$, then

$$e^{-B} = \frac{e^{-\mu_1}}{\mu_1 - \mu_2}(B - \mu_2 I) + \frac{e^{-\mu_2}}{\mu_2 - \mu_1}(B - \mu_1 I)$$

where I is the 2×2 identity matrix. It follows that

$$\mathbf{e}_1^T \left(e^{-B}\right) \mathbf{e}_1 = \frac{b_{11}(e^{-\mu_1} - e^{-\mu_2}) + \mu_1 e^{-\mu_2} - \mu_2 e^{\mu_1}}{\mu_1 - \mu_2}.$$

Next, we build explicitly the matrix J_2 and compute its eigenvalues. The values of $\omega_1 = -\mu$ and $\gamma_1 = \sigma$ are derived applying one step of Lanczos iteration to the matrix $-A$ with starting vectors $\mathbf{x}_{-1} = \mathbf{0}$ and $\mathbf{x}_0 = \frac{1}{\sqrt{n}}\mathbf{1}$. We want to compute the value of ω_2 in such a way that the matrix J_2 has the prescribed eigenvalue $\tau_1 = \alpha$ or $\tau_1 = \beta$. Note that $\gamma_1 = 0$ if and only if the graph is regular. In such case we simply take $\omega_2 = \tau_1$ and the matrix J_2 is diagonal with eigenvalues $\mu_1 = -\mu$ and $\mu_2 = \tau_1$. Thus, let us assume $\gamma_1 \neq 0$. In order to compute the value for ω_2, we use the three–term recurrence for orthogonal polynomials:

$$\gamma_j p_j(\lambda) = (\lambda - \omega_j) p_{j-1}(\lambda) - \gamma_{j-1} p_{j-2}(\lambda), \quad j = 1, 2, \ldots, p,$$

with $p_{-1}(\lambda) \equiv 0, p_0(\lambda) \equiv 1$ to impose that $p_2(\tau_1) = 0$ and hence derive $\omega_2 = \tau_1 - \frac{\gamma_1}{p_1(\tau_1)}$. Using the same recurrence, we also find that $p_1(\tau_1) = \frac{\tau_1 - \omega_1}{\gamma_1}$ which is nonzero, since the zeros of orthogonal polynomials satisfying the three–term recurrence are distinct and lie in the interior of $[\alpha, \beta]$ (see [22, Theorem 2.14]).

Finally, the matrix

$$J_2 = \begin{pmatrix} \omega_1 & \gamma_1 \\ \gamma_1 & \tau_1 - \frac{\gamma_1^2}{\tau_1 - \omega_1} \end{pmatrix}$$

has (distinct) eigenvalues $\mu_1 = \tau_1$ and $\mu_2 = \omega_1 + \frac{\gamma_1^2}{\omega_1 - \tau_1}$. This, together with Theorem 3.1 and the relation (3.3), concludes the proof. □

Following the same procedure, analogous bounds can be found for the adjacency matrix of the graph after performing a downdate or an update. These results are summarized in the following Corollaries.

COROLLARY 1. *[Downdating] Let $\widehat{A} = A - UW^T$, where $U = [\mathbf{e}_i, \mathbf{e}_j]$ and $W = [\mathbf{e}_j, \mathbf{e}_i]$ be the adjacency matrix of an unweighted and undirected network obtained after the downdate of the edge (i, j) from the matrix A. Let $\omega_1 = -\mu = -\frac{1}{n}\sum_{i=1}^n d_i$ and $\gamma_1 = \sigma = \sqrt{\frac{1}{n}\sum_{k=1}^n (d_k - \mu)^2}$, where d_i is the degree of node i in the original graph. Then*

$$\Phi\left(\beta_-, \omega_- + \frac{\gamma_-^2}{\omega_- - \beta_-}\right) \leq \frac{TC(\widehat{A})}{n} \leq \Phi\left(\alpha_-, \omega_- + \frac{\gamma_-^2}{\omega_- - \alpha_-}\right)$$

where

$$\begin{cases} \omega_- = \omega_1 + \frac{2}{n}; \\ \gamma_- = \sqrt{\gamma_1^2 - \frac{2}{n}\left(d_i + d_j - 1 + 2\omega_1 + \frac{2}{n}\right)} \end{cases},$$

α_- and β_- are approximation of the smallest and largest eigenvalues of $-\widehat{A}$ respectively, and Φ is defined as in equation (3.4) with $c = \omega_-$.

Note that if bounds α and β for the extremal eigenvalues of the original matrix are known, we can then use $\alpha_- = \alpha$ and $\beta_- = \beta + 1$. Indeed, if we order the eigenvalues of $\widehat{A}$ in non–increasing order $\widehat{\lambda}_1 > \widehat{\lambda}_2 \geq \ldots \geq \widehat{\lambda}_n$ we obtain as a consequence of Weyl's Theorem (see [27, Section 4.3]) that

$$\alpha - 1 \leq -\lambda_1 - 1 < -\widehat{\lambda}_1 < -\widehat{\lambda}_2 \leq \ldots \leq -\widehat{\lambda}_n < -\lambda_n + 1 \leq \beta + 1.$$

Furthermore, the Perron–Frobenius Theorem ensures that, when performing a downdate, the largest eigenvalue of the adjacency matrix cannot increase; hence, we deduce the more stringent bounds $\alpha \leq -\widehat{\lambda}_1 \leq -\widehat{\lambda}_2 \leq \ldots \leq -\widehat{\lambda}_n \leq \beta + 1$.

Similarly, we can derive bounds for the normalized total communicability of the matrix $\tilde{A}$ obtained from the matrix A after performing the update of the virtual edge (i, j).

COROLLARY 2. *[Updating] Let $\tilde{A} = A + UW^T$, where $U = [\mathbf{e}_i, \mathbf{e}_j]$ and $W = [\mathbf{e}_j, \mathbf{e}_i]$ be the adjacency matrix of an unweighted and undirected network obtained after the update of the virtual edge (i, j) in the matrix A. Let $\omega_1 = -\mu = -\frac{1}{n}\sum_{i=1}^n d_i$ and $\gamma_1 = \sigma = \sqrt{\frac{1}{n}\sum_{k=1}^n (d_k - \mu)^2}$, where d_i is the degree of node i in the original graph. Then*

$$\Phi\left(\beta_+, \omega_+ + \frac{\gamma_+^2}{\omega_+ - \beta_+}\right) \leq \frac{TC(\tilde{A})}{n} \leq \Phi\left(\alpha_+, \omega_+ + \frac{\gamma_+^2}{\omega_+ - \alpha_+}\right)$$

where

$$\begin{cases} \omega_+ = \omega_1 - \frac{2}{n}; \\ \gamma_+ = \sqrt{\gamma_1^2 + \frac{2}{n}\left(d_i + d_j + 1 + 2\omega_1 - \frac{2}{n}\right)} \end{cases},$$

α_+ and β_+ are bounds for the smallest and largest eigenvalues of $-\tilde{A}$ respectively, and Φ is defined as in equation (3.4) with $c = \omega_+$.

Notice that again, if bounds α and β for the extremal eigenvalues of $-A$ are known, we can then take $\alpha_+ = \alpha - 1$ and $\beta_+ = \beta$. In fact, the spectrum of the rank-two symmetric perturbations UW^T and $-UW^T$ is $\{\pm 1, 0\}$ and hence we can use Weyl's Theorem as before and then improve the upper bound using the Perron–Frobenius Theorem.

In the next section we will see how the new bounds can be used to guide the updating and downdating process.

4. Modifications of the adjacency matrix. In this section we develop techniques that allow us to tackle the following problems.

(P1) Downdate: select K edges that can be downdated from the network without disconnecting it and that cause the smallest drop in the total communicability of the graph;

(P2) Update: select K edges to be added to the network (without creating self–loops or multiple edges) so as to increase as much as possible the total communicability of the graph;

(P3) Rewire: select K edges to be rewired in the network so as to increase as much as possible the value of $TC(A)$. The rewiring process must not disconnect the network or create self–loops or multiple edges in the graph.

As we will show below, (P3) can be solved using combinations of methods developed to solve (P1) and (P2). Hence, we first focus on the downdate and the update separately. Note that to decrease as little as possible the total communicability when removing an edge it would suffice to select $(i^*, j^*) \in E$ so as to minimize the quantities

$$\mathbf{1}^T A^k \mathbf{1} - \mathbf{1}^T (A - UW^T)^k \mathbf{1} \qquad \forall k = 1, 2, \ldots,$$

since $TC(A) = \sum_{k=0}^{\infty} \frac{\mathbf{1}^T A^k \mathbf{1}}{k!}$.

Similarly, to increase as much as possible $TC(A)$ by addition of a virtual edge, it would suffice to select $(i^*, j^*) \in \overline{E}$ that maximizes the differences

$$\mathbf{1}^T (A + UW^T)^k \mathbf{1} - \mathbf{1}^T A^k \mathbf{1} \qquad \forall k = 1, 2, \ldots$$

However, it is easy to show that in general one can not find a choice for (i^*, j^*) that works for all such k. Indeed, numerical experiments on small synthetic graphs (not shown here) show that in general the optimal edge selection for $k = 2$ is different from the one for $k = 3$. Because of this, it is unlikely that one can find a simple "closed form solution" to the problem, and we need to develop approximation techniques.

At this point we introduce three new definitions of edge centrality.

DEFINITION 4.1. *For any $i, j \in V$ we define the* edge subgraph centrality *of an existing/virtual edge (i, j) as*

$$^eSC(i, j) = \left(e^A\right)_{ii} \left(e^A\right)_{jj}. \tag{4.1}$$

DEFINITION 4.2. *For any $i, j \in V$ we define the* edge eigenvector centrality *of an existing/virtual edge (i, j) as*

$$^{e}EC(i,j) = q_1(i)q_1(j). \tag{4.2}$$

DEFINITION 4.3. *For any $i, j \in V$ we define the* edge total communicability centrality *of an existing/virtual edge (i, j) as*

$$^{e}TC(i,j) = [e^A\mathbf{1}]_i[e^A\mathbf{1}]_j. \tag{4.3}$$

It is important to observe that when the spectral gap $\lambda_1 - \lambda_2$ is "large enough", then both the subgraph centrality $\left(e^A\right)_{ii}$ and the total communicability centrality $[e^A\mathbf{1}]_i$ are essentially determined by $e^{\lambda_1}q_1(i)^2$, see, e.g., [6, 7, 15]; it follows that in this case the three definitions can be expected to provide similar rankings of the edges, as is the case for the ranking of nodes. This is especially true when attention is restricted to the top edges (or nodes).

It is worth noting that the definition of the centrality of an edge based on the centralities of its incident nodes is not a new idea in literature (see, e.g., [8]). Moreover, note that we defined these measures of centrality for both existing and virtual edges (as in [8]). The reason for this as well as the justification for these definitions will become clear in the next subsections.

(P1) Downdate. The downdate of any edge in the network will result in a reduction of its total communicability. Since the total communicability is an index of the ease of sending information across the network, our main goal when performing a downdate is to reduce as little as possible the decrease in this index. Note that since we are focusing on the case of connected networks, we will only perform downdates that keep the resulting graph connected. In practice, it is desirable to further restrict the choice of downdates to a subset of all existing edges, on the basis of criteria to be discussed shortly.

An "optimal" approach would select at each step of the downdating process a candidate edge corresponding to the minimum decrease of communicability.[1] Note that for large networks this method is too costly to be practical. For this reason we aim to develop inexpensive techniques that will hopefully give close–to–optimal results. Nevertheless, for small networks we will use the "optimal" approach (where we systematically try all feasible edges and delete the one causing the least drop in total communicability) as a baseline method against which we compare the various algorithms discussed below. This method will be henceforth referred to as `optimal`.

The next method we introduce performs the downdate of the lowest ranked existing edge according to the edge subgraph centrality whose removal does not disconnect the network. We will refer to this method as `subgraph`. From the point of view of the communicability, `subgraph` downdates an edge connecting two nodes which are peripheral (i.e., have low centrality) and therefore are not expected to give a large contribution to the spread of information along the network. As further justification

[1] Strictly speaking, this would correspond to a greedy algorithm which is only locally optimal. In general, this is unlikely to result in "globally optimal" network communicability. In this paper, the term "optimal" will be understood in this limited sense only.

for this approach, we observe that

$$\left(e^A\right)_{ii}\left(e^A\right)_{jj} > \left(e^A\right)_{ij}^2 \qquad \forall i,j \in V, \tag{4.4}$$

since e^A is positive definite. Hence, the selected edge is connecting two nodes whose ability to exchange information is already very low, and we do not expect the total communicability to suffer too much from this edge removal. This observation also suggests that such downdates can be repeatedly applied without the need to recompute the ranking of the edges after each downdate. As long as the number of downdates performed remain small compared to the total number of edges, we expect good results at a greatly reduced total cost. Note also that such downdates can be performed simultaneously rather than sequentially. We will refer to this variant as `subgraph.no`.

The third method we introduce is based on Definition 4.2 and consists of removing at each step the existing edge (i,j) for which the product $q_1(i)q_1(j)$ is minimum and whose removal does not disconnect the graph. The motivation for this strategy is that, as mentioned, for graphs with sufficiently large spectral gap $\lambda_1 - \lambda_2$ the node centrality rankings produced by $(e^A)_{ii}$ are close to those obtained using the eigenvector centralities $q_1(i)$. In other words, when the gap is large, the rank-one approximation $e^A \approx e^{\lambda_1}\mathbf{q}_1\mathbf{q}_1^T$ is pretty good. Observe that computing the eigenvector centralities is less expensive than computing the subgraph centralities. Again, this strategy can be applied with or without recomputing the rankings (i.e., the dominant eigenvector $\mathbf{q}_1$) after each downdate. We will refer to these two strategies (with and without the recalculation of edge centralities) as `eigenvector` and `eigenvector.no`, respectively.

Yet another method can be obtained on the basis of Definition 4.3 and consists of removing at each step the existing edge (i,j) for which the product of the total node communicabilities $[e^A\mathbf{1}]_i$ and $[e^A\mathbf{1}]_j$ is minimum. This method can be seen as intermediate between the methods `subgraph` and `eigenvector`. Like these, it can be implemented with or without recalculating the edge centralities after each downdate; we refer to these two variants as `nodeTC` and `nodeTC.no`. It is expected again that the results obtained with these methods will be close to those obtained using `subgraph` and `subgraph.no` if the spectral gap of A is sufficiently large [6].

Finally, we consider a technique motivated by the bounds obtained via quadrature rules derived in section 3. From the expression for the function Φ in the special case of the downdate (cf. Corollary 1), we infer that a potentially good choice may be to remove the edge having incident nodes i,j for which the sum $d_i + d_j$ is minimal, if its removal does not disconnect the network. Indeed, this choice reduces the upper bound only slightly and the total communicability may mirror this behavior. Another way to justify this strategy is to observe that it is indeed the optimal strategy if we approximate e^A with its second-order approximation $I + A + \frac{1}{2}A^2$ in the definition of total communicability. This technique will be henceforth referred to as `degree`.

We note that a related measure, namely, the average of the out-degrees $\frac{d_i+d_j}{2}$, was proposed in [8] as a measure for the centrality of an edge (i,j) in directed graphs.

(P2) Update. Most real world networks are characterized by low average degree. As a consequence, the adjacency matrices of such networks are sparse ($m = O(n)$). For the purpose of selecting a virtual edge to be updated, this implies that we have approximately $\frac{1}{2}\left(n^2 - cn\right)$ possible choices if we want to avoid the formation of multiple edges or self–loops (here c is a moderate constant). Each one of these possible updates will result in an increase of the total communicability of the network, but not every one of these will result in a significant increment.

One natural updating technique is to connect two nodes having high subgraph centralities, i.e., add the virtual edge having the highest ranking according to the edge subgraph centrality ${}^{e}SC$. Its incident nodes, being quite central, can be expected to have an important role in the spreading of information along the network; on the other hand, the communication between them may be relatively poor (think for example of the case where the two nodes sit in two distinct communities). Hence, giving them a preferential communication channel, such as an edge between them, should result in a better spread of information along the whole network. Again, we will use the label `subgraph` to describe this updating strategy. As before, in order to reduce the computational cost, we also test the effectiveness of this technique without the recomputation of the ranking of the virtual edges after each update. This variant (referred to as `subgraph.no`) is expected to return good results as well, since the selected update should not radically change the ranking of the edges. Indeed, it makes central nodes even more central, and the ranking of the edges remains consequently almost unchanged. Note again that these updates can be performed simultaneously rather than sequentially.

As for the case of downdating, less expensive methods result if eigenvector centrality or node total communicability are used instead of subgraph centrality. Hence, we also consider adding the virtual edge having the largest edge eigenvector or edge total communicability centrality. We expect the corresponding results to be comparable to those obtained with edge subgraph centrality when the spectral gap is not too small. As before, the corresponding updating techniques (with and without recalculation of the node centralities) will be referred to as `eigenvector`/`nodeTC` and `eigenvector.no`/`nodeTC.no`, respectively.

Finally, the bounds via quadrature rules derived in section 3 suggest adding the virtual edge (i, j) for which $d_i + d_j$ is maximal. Indeed, such a choice would maximize the lower bound on the total communicability, see Corollary 2. Again, this choice can also be justified by noting that it is optimal if e^A is replaced by its quadratic Maclaurin approximant. We will again use the label `degree` to refer to this updating strategy.

All these techniques will be compared with the `optimal` one, based on systematically trying all feasible virtual edges and selecting at each step the one resulting in the largest increase of the total communicability. Due to the very high cost of this brute force approach, we will use it only on small networks.

(P3) Rewire. As we have already noted, there are situations in which the rewire of an edge may be preferable to the addition of a new one. There are various possible choices for the rewiring strategy to follow. The greatest part of those found in literature are variants of random rewiring (see for example [9, 29]). In this paper, on the other hand, we are interested in devising mathematically informed rewiring strategies. Therefore, we will compare our rewiring methods to the random rewire method, `random`, which downdates an edge (chosen uniformly at random among all edges whose removal does not disconnect the network) and then updates a virtual edge, also chosen uniformly at random.

Combining the various downdating and updating methods previously introduced we obtain different rewiring strategies based on the centralities of edges and on the bounds for the total communicability. Concerning the methods based on the edge subgraph, eigenvector, and total communicability centrality, we note that since a single downdate does not dramatically change the communication capability of the network, we do not need to recompute the centralities and the ranking of the edges after each

downdating step, at least as long as the number of rewired edges remains relatively small (numerical experiments not shown here support this claim). On the other hand, after each update we may or may not recalculate the edge centralities. As before, we use `subgraph/subgraph.no`, `eigenvector/eigenvector.no` and `nodeTC/nodeTC.no` to refer to these three variants of rewiring. Additionally, we introduce another rewiring strategy, henceforth referred to as `node`, based on the subgraph centrality of the nodes. In this method we disconnect the most central node from the least central node among its immediate neighbors; then we connect it to the most central node among those it is not linked to. It is worth emphasizing that this strategy is philosophically different from the previous ones based on the edge subgraph centrality in the downdating phase (the updating step is the same). In fact, in those methods we use information on the nodes in order to deduce some information on the edges connecting them; on the other hand, the `node` algorithm does not take into account the potentially high "payload" of the edges involved, whose removal may result in a dramatic drop in the total communicability.

4.1. Computational aspects. There are several important points to keep in mind when implementing the methods described in the previous subsection. First of all, for the downdates, updates, and rewires based on the edge subgraph centrality we need to compute the diagonal entries of e^A. This is the most expensive part of these methods. There are, however, techniques that can be used to rapidly estimate the diagonal entries of e^A and to quickly identify the top ℓ nodes, where $\ell \ll n$; see [4, 6, 20] and references therein. It should be pointed out that very high accuracy is not required or warranted by the problem. We also recall that the same techniques (based on quadrature rules and the Lanczos process) can be used to compute the total communicability $\mathbf{1}^T e^A \mathbf{1}$ quickly (typically in $O(n)$ work), although such computation is actually not required by any of the algorithms tested here except by the `optimal` strategy, which is only used (for small networks) as a baseline method. Such methods can also be used for rapidly estimating the node total communicability centralities, $TC(i) = [e^A \mathbf{1}]_i = \mathbf{e}_i^T e^A \mathbf{1}$.

Secondly, when performing an update or the updating phase of a rewire, it makes sense to work with a subset of the set of all virtual edges $\overline{E}$. Indeed, for a sparse network $\overline{E}$ contains $O(n^2)$ edges and for large n this is prohibitive. Due to the particular selection criteria we want to use, it is reasonable to restrict ourselves to the virtual edges in the subgraph of our network that are incident to a subset S of nodes containing a certain percentage of the top nodes, ranked according to some centrality measure. We found that for the larger networks considered in this paper, using just the top 1% of the nodes ranked using eigenvector centrality yields very good results.

Next, we derive the computational costs for the downdating techniques used in this paper. Let m be the number of edges in the network and let $K \ll n$, assumed bounded independently of n as $n \to \infty$, be the maximum number of downdates we want to perform; in this paper, the maximum value of K we consider is 2000 (used for the three largest networks in our data set).

In the `optimal` method we remove each edge in turn, compute the total communicability after each downdate, and then choose the downdate which caused the least decrease in $TC(A)$; assuming that the cost of computing $TC(A)$ is $O(n)$, we find a total cost of $O(Kmn)$ for K updates. Since $m = O(n)$, this amounts to $O(Kn^2)$.

Next, we consider the cost of techniques based on subgraph centralities. The cost of computing the node subgraph centralities is not easy to assess in general, since it depends on network properties and on the approximation technique used. If a

rank-k approximation is used [20], the cost is approximately $O(kn)$; hence, the cost is linear in n if k is independent of n, which is appropriate for many types of networks. Computing the edge centralities requires another $m = O(n)$ operations, and sorting the edges by their centralities costs approximately $m \ln m$ comparisons. Note that sorting is only necessary in the `subgraph.no` variant of the algorithm; indeed, with `subgraph` we recompute the centralities after each update and instead of sorting the result we only need to identify the edge of minimum centrality at each step, which can be done in $O(m)$ work. Summarizing, the cost of `subgraph` is $O(K(n+m)) = O(Kn)$ for K downdates if we assume the subgraph centralities can be computed in $O(n)$ time, and the cost for `subgraph.no` is $O(n+m) = O(n)$ plus a pre-processing cost of $O(m \ln m)$ $(= O(n \ln n))$ comparisons for sorting the edge centralities. Although the asymptotic cost of `subgraph.no` appears to be higher than that of `subgraph` (due to the $n \ln n$ term), in practice one finds that `subgraph.no` runs invariably much faster than `subgraph` for all cases tested here.

The costs associated with `eigenvector` and `eigenvector.no` scale like those of `subgraph` and `subgraph.no`, assuming that the dominant eigenvector $\mathbf{q}_1$ of a large sparse $n \times n$ adjacency matrix can be approximated in $O(n)$ time. For many real world networks this is a reasonable assumption, since in practice we found that running a fixed number of Lanczos steps will give a sufficiently good approximation of $\mathbf{q}_1$. The prefactors can be expected to be much smaller for the methods based on eigenvector centrality than for those based on subgraph centrality.

The costs for `nodeTC` and `nodeTC.no` are comparable to those for `eigenvector` and `eigenvector.no`, with the same asymptotic scalability.

Finally, the cost of `degree` is $O(Km)$ and hence also $O(Kn)$ for a sparse network.

Note that the cost of checking that the connectivity is preserved after each downdate does not affect these asymptotic estimates; indeed, using A^* search [24] this can be done in $O(m)$ time and hence the additional cost is only $O(n)$ for a sparse network. Of course, if the removal of an edge is found to disconnect the network we do not perfom the downdate and move on to the next candidate edge.

We consider next the computational cost for the updating strategies. As before we let K, assumed bounded independently of n as $n \to \infty$, be the maximum number of updates we want to perform.

It can be easily shown that the `optimal` method costs $O(Kn^3)$ operations. To estimate the cost of the remaining methods, we assume that the set $S \subset V$ consisting of the top $\ell = |S|$ nodes (ranked according to some centrality measure) is known. The cost of determining this set is asymptotically dominated by the term $O(n \ln n)$, as we saw. As already mentioned, ℓ will be equal to some fixed percentage of the total number of nodes in the network.

Both `subgraph` and `eigenvector` cost $O(K\ell n)$ operations, provided a low rank approximation (of fixed rank) is used to estimate the subgraph centralities. The same holds for `nodeTC`. Typically, the prefactor will be larger for the former method. Since we assumed that $\ell = O(n)$ (albeit with a very small prefactor, like 10^{-2}) these methods exhibit an $O(n^2)$ scaling. In practice this is somewhat misleading, since the quadratic scaling is not observed until n is quite large.

Finally, `degree` costs $O(K\ell) = O(n)$ while `subgraph.no`, `eigenvector.no` and `nodeTC.no` all cost $O((K+\ell)n)$. Again, the latter cost is asymptotically quadratic but the actual cost is dominated by the linear part until n becomes quite large. We note that we can obtain an asymptotically linear scaling by imposing an upper bound on ℓ, i.e., on the fraction of nodes that we are willing to include in the working subset

TABLE 1
Description of the Data Set.

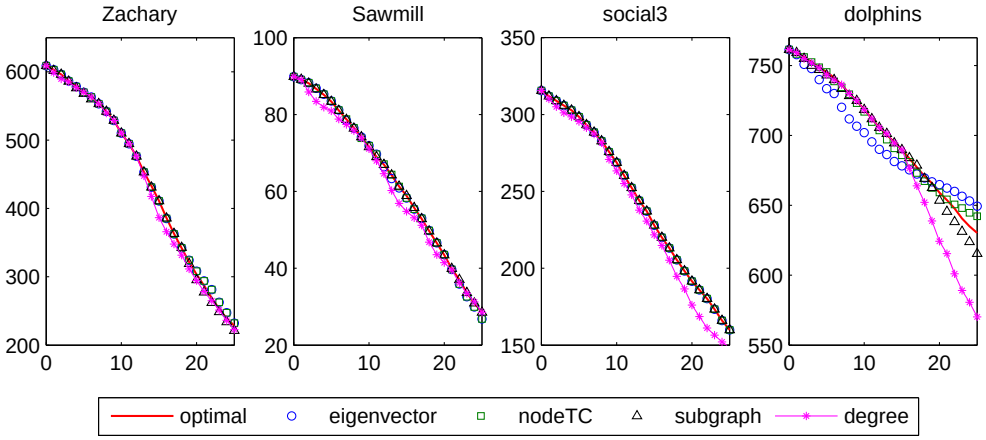
NAME	n	m	λ_1	λ_2	$\lambda_1 - \lambda_2$
Zachary	34	78	6.726	4.977	1.749
Sawmill	36	62	4.972	3.271	1.701
social3	32	80	5.971	3.810	2.161
dolphins	62	159	7.193	5.936	1.257
Minnesota	2640	3302	3.2324	3.2319	0.0005
USAir97	332	2126	41.233	17.308	23.925
as–735	6474	12572	46.893	27.823	19.070
Erdös02	5534	8472	25.842	12.330	13.512
ca–HepTh	8638	24806	31.034	23.004	8.031
as–22july2006	22963	48436	71.613	53.166	18.447
usroad–48	126146	161950	3.911	3.840	0.071

S of nodes. We stress that because of the widely different prefactors for the various methods, these asymptotic estimates should only be taken as indicative. In the next section we present timings showing the linear scaling behavior of the various heuristics in practice, at least for the networks considered here.

5. Numerical studies. In this section we discuss the results of numerical studies performed in order to assess the effectiveness and efficiency of the proposed techniques. The tests have been performed on both synthetic and real-world networks, as described below.

5.1. Real–world networks. The real-world networks used in the tests (see Table 1) come from a variety of sources. The Zachary Karate Club network is a classic example in social network analysis [43]. The Sawmill and social3 networks were provided to us by Prof. Ernesto Estrada. The Sawmill network describes a communication network within a small enterprise (see [33, 35]), whereas social3 is a network of social contacts among college students participating in a leadership course (see [44]). All the other networks can be found in the University of Florida Sparse Matrix Collection [13] under different "groups". The network dolphins (see [30]) is in the Newman group and represents a social network of frequent associations between 62 dolphins in a community living in the waters off New Zealand. The USAir97 and Erdös02 networks are from the Pajek group. The USAir97 network describes the US Air flight routes in 1997, while the Erdös02 network represents the Erdös collaboration network, Erdös included. The network as–735, from the SNAP group, is the communication network of a group of autonomous system (AS) measured over 735 days between November 8, 1997 and January 2, 2000. Communication occurs when routers from two ASs exchange information. The Minnesota network from the Gleich group represents the Minnesota road network. These latter three networks are not connected, therefore the tests were performed on their largest connected component. We point out that the original largest connected component of the network as–735 has 1323 ones on the main diagonal which were retained in our tests. The network ca–HepTh is from the SNAP group and represents the collaboration network of arXiv High Energy Physics Theory; the network as–22july06 is from the Newman group and represents the (symmetrized) structure of Internet routers as of July 22, 2006. Finally, the network usroad–48, which is from the Gleich group, represents the continental US road network. For each network, Table 1 reports the number of nodes (n), the number of edges (m), the two largest eigenvalues, and the spectral gap. We use the first eight networks to test all methods described in the previous section (except for `optimal`,

FIG. 1. *Evolution of the normalized total communicability vs number of downdates performed on small networks.*



which is only applied to the four smallest networks) and the last three to illustrate the performance of the most efficient among the methods tested.

We begin by showing results for the four smallest networks. Figure 1 displays the results obtained with the downdating methods `optimal`, `eigenvector`, `nodeTC`, `subgraph`, and `degree`. The results for `eigenvector.no`, `subgraph.no`, and `nodeTC.no` are virtually indistinguishable from those obtained with `eigenvector`, `subgraph` and `node` and are therefore not shown. At each step we modify the network by downdating an edge and we then compute and plot the new value of the normalized total communicability. The tests consist of 25 modifications.

Figure 1 shows that our methods all perform similarly and give results that are in most cases very close to those obtained with `optimal`, and occasionally even better, as is the case for `eigenvector` (and `eigenvector.no`) on the dolphins network after a sufficient number of downdates have been performed. This result may seem puzzling at first, however, it can be easily explained by noticing that `eigenvector` selects a different edge from that selected by `optimal` at the third downdate step. Hence, from that point on the adjacency matrices on which the methods work are different, and the choice performed by the `optimal` method may no longer be optimal for the graph manipulated by `eigenvector`. Note that even the simple heuristic `degree` seems to perform well, except perhaps on the dolphins network after 15 or so downdate steps. Overall, the methods based on eigenvector and total communicability centrality appear to perform best in view of their efficacy and low cost.

The results for the updating methods are reported in Figure 2. As for the downdating methods, `subgraph.no`, `eigenvector.no`, and `nodeTC.no` return results that are virtually identical to those obtained using `subgraph`, `eigenvector` and `nodeTC`, therefore we omit them from the figure. Once again we see that the methods based on eigenvector, subgraph, and total communicability centrality give excellent results, whereas `degree` is generally not as effective.

Rewiring results are displayed in Figure 3. Clearly, the methods making use of edge centrality perform quite well, in contrast to `random` rewiring (which is only included as a base for comparison). Note also the poor performance of `node`, showing that the use of edge centralities (as opposed to node centralities alone) is indispensable in this context.

FIG. 2. *Evolution of the normalized total communicability vs number of updates performed on small networks.*

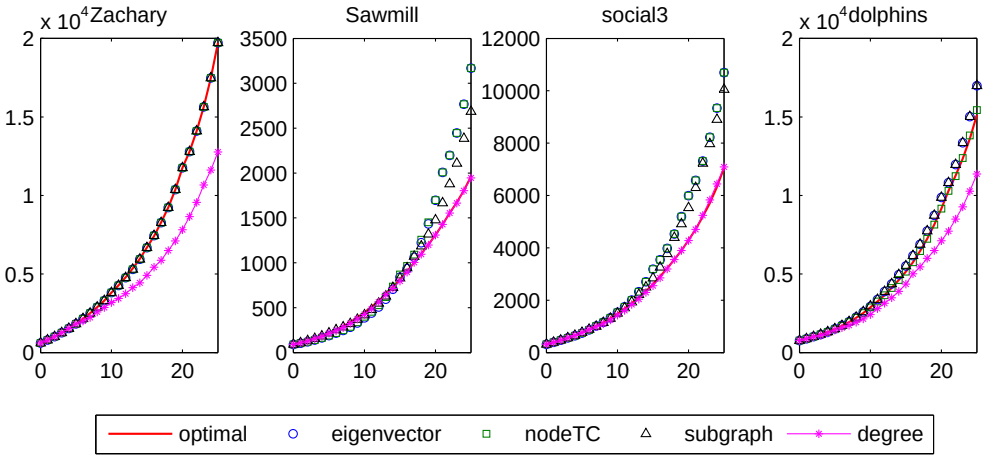
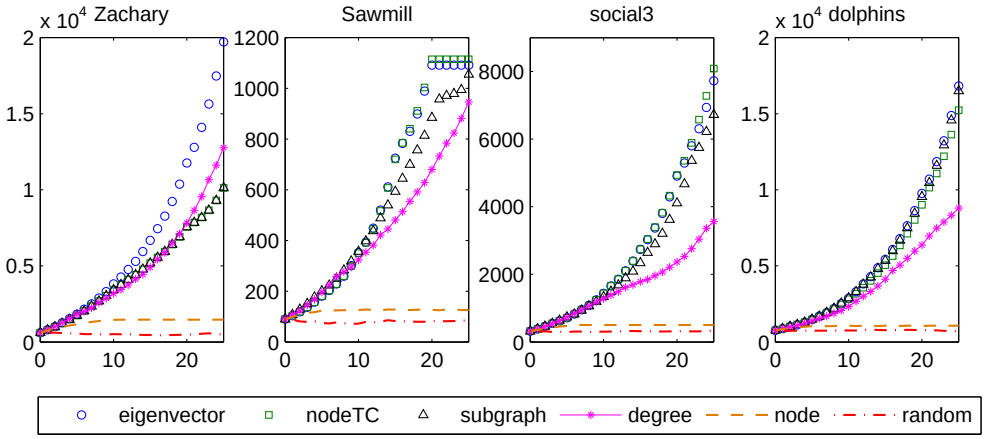
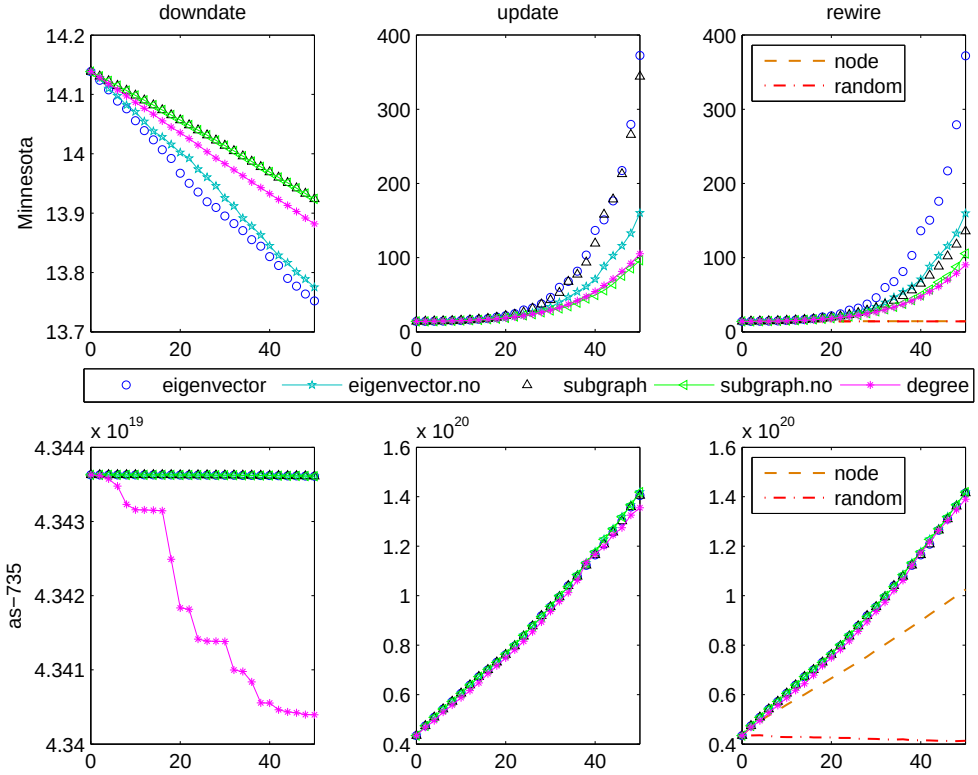


FIG. 3. *Evolution of the normalized total communicability vs number of rewires performed on small networks.*



Next, we consider the medium size networks (Minnesota, as735, USAir97, and Erdös02). For these networks the set $\overline{E}$ (the complement of the set E of edges) is large enough that performing an extensive search for the edge to be updated is expensive. Hence, we form the set S of the top 10% of the nodes ordered according to the eigenvector centrality and we restrict our search to virtual edges incident to these nodes only. An exception is the network USAir97 where we have used the set S corresponding to the top 20% of the nodes, since in the case of 10% this set contained only 52 virtual edges. In Figures 4 and 5 we show results for the methods `eigenvector`, `eigenvector.no`, `subgraph`, `subgraph.no` and `degree`. These results confirm the effectiveness of the `eigenvector` and `subgraph` algorithms and their less expensive variants `eigenvector.no` and `subgraph.no` in nearly all cases; similar results were obtained with `nodeTC` and `nodeTC.no` (not shown). The only exception is in the downdating of the Minnesota network, where the eigenvector-based techniques give slightly worse results. This fact is easily explained in view of the tiny spectral

FIG. 4. *Evolution of the normalized total communicability vs number of downdates, updates and rewires for networks Minnesota and as735.*



gap characterizing this and similar networks[2] (see Table 1). Because of this property, eigenvector centrality is a poor approximation of subgraph centrality and cannot be expected to give results similar to those obtained with `subgraph` and `subgraph.no`.

The results also show that the inexpensive `degree` method does not perform as well on these networks, except perhaps on Minnesota. The relatively poor performance of this method is due to the fact that the information used by this method to select an edge for downdating is too local.

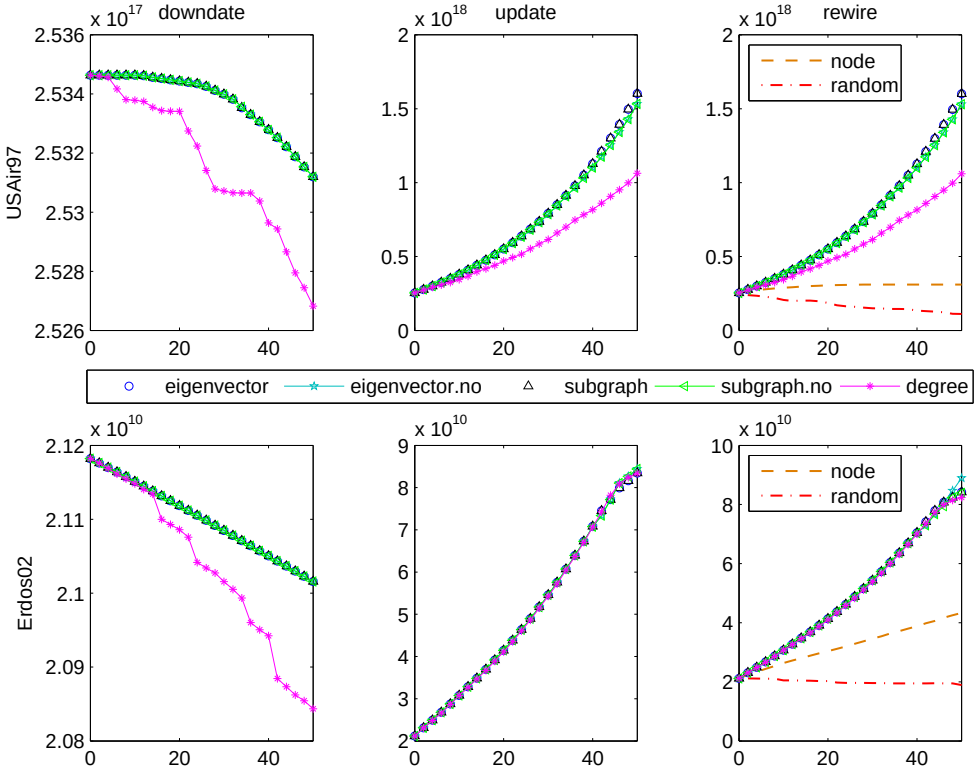
Note, however, the scale on the vertical axis in Figures 4–5, suggesting that for these networks (excluding perhaps Minnesota) all the edge centrality-based methods perform well with only very small relative differences between the resulting total communicabilities.

Overall, these results indicate that the edge centrality-based methods, especially the inexpensive `eigenvector.no` and `nodeTC.no` variants, are an excellent choice in almost all cases. In the case of downdating networks with small spectral gaps, `subgraph.no` may be preferable but at a higher cost.

The behavior of the `degree` method depends strongly on the network on which it is used. Our tests indicate that it behaves well in some cases (for example, Erdös02) but poorly in others (Minnesota). We speculate that this method may perform adequately

[2]Small spectral gaps are typical of large, grid-like networks such as the road networks or the graphs corresponding to triangulation or discretization of physical domains.

FIG. 5. *Evolution of the normalized total communicability vs number of downdates, updates and rewires for networks USAir97 and Erdös02.*



on scale-free networks (such as Erdos02) where a high degree is an indication of centrality in spreading information across the network.

Some comments on the difference in the results for updating as compared to those for rewiring (downdating followed by updating) are in order. Recall that our downdating strategies aim to reduce as little as possible the decrease in the value of the total communicability, whereas the updating techniques aim to increase this index as much as possible. With this in mind, it is not surprising to see that the trends of the evolution of the total communicability after rewiring reflect those obtained with the updating strategies. The values obtained using the updates are in general higher than those obtained using the rewiring strategies, since updating implies the addition of edges whereas in rewiring the number of edges remains the same. The difference is especially pronounced for the small networks (except for dolphins), where the effects of downdates has a greater impact, leading to a decrease of up nearly 70% of the original value of the total communicability after 25 downdates (cf. Figure 1). It is important to stress that the methods based on the edge eigenvector and total communicability centrality appear to be more stable than the others under rewiring and to dampen the effect of the downdates even for small networks.

Finally, in Figures 6-7 we show results for the three largest networks in our data set (ca–HepTh, as-22july06 and usroad-48). Here we compare the following methods: `eigenvector`, `eigenvector.no`, `nodeTC`, `nodeTC.no`, `subgraph.no` and `degree`; random downdating was also tested and found to give poor results. Note that network

FIG. 6. *Downdates for large networks.*

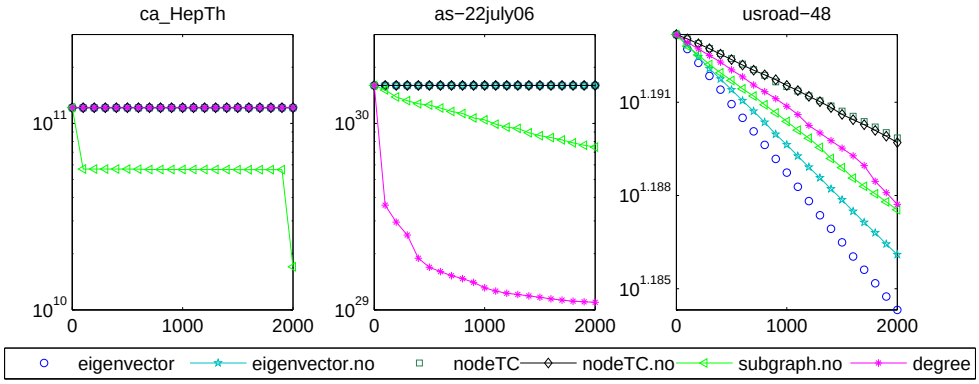
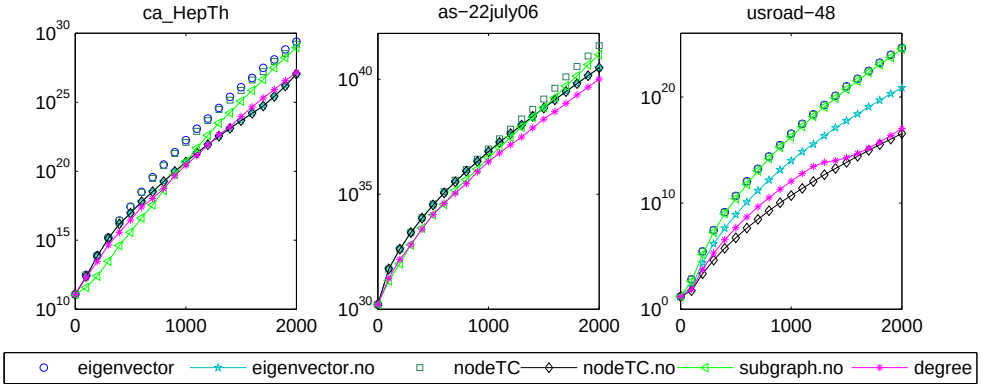


FIG. 7. *Updates for large networks.*

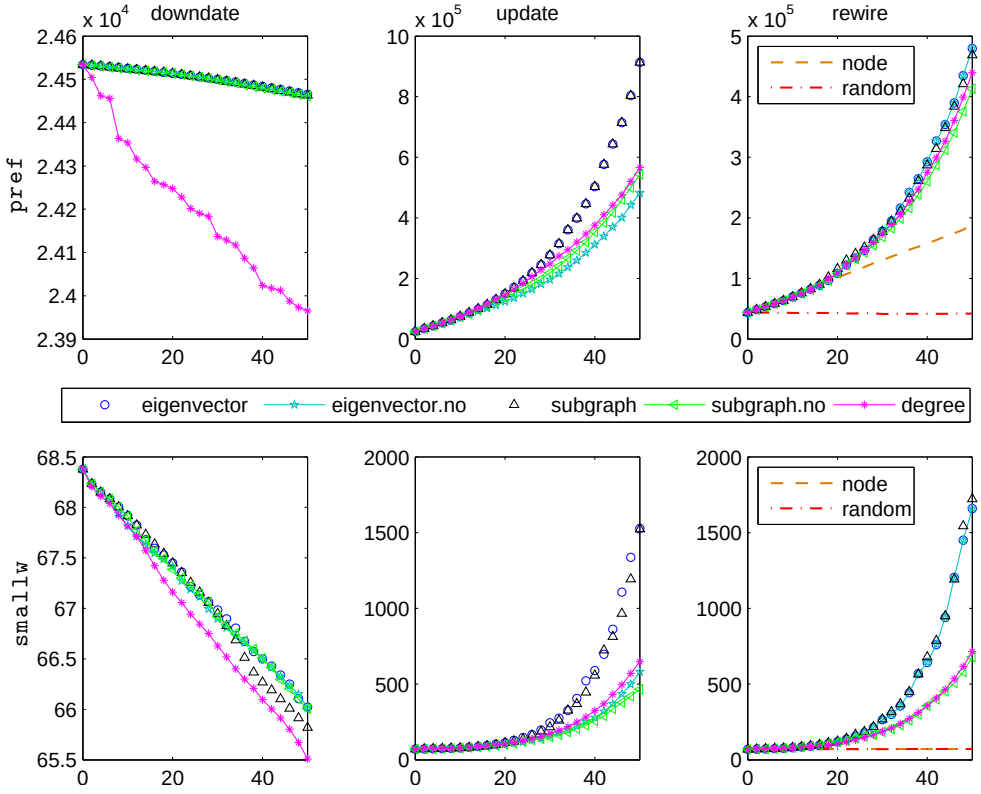


usroad-48 behaves similarly to Minnesota; this is not surprising in view of the fact that these are both road networks with a tiny spectral gap. Looking at the scale on the vertical axis, however, it is clear that the decrease in total communicability is negligible with all the methods tested here. The results on these networks confirm the general trend observed so far; in particular, we note the excellent behavior of `nodeTC` and `nodeTC.no`.

5.2. Synthetic networks. The synthetic examples used in the tests were produced using the CONTEST toolbox for Matlab (see [39, 40]). We tested two types of graphs: the preferential attachment (Barabási–Albert) model and the small world (Watts–Strogatz) model.

The preferential attachment model [3] was designed to produce networks with scale–free degree distributions as well as the small world property, characterized by short average path length and relatively high clustering coefficient. In CONTEST, preferential attachment networks are constructed using the command `pref(n,d)` where n is the number of nodes and $d \geq 1$ is the number of edges each new node is given when it is first introduced to the network. The network is created by adding nodes one by one (each new node with d edges). The edges of the new node connect

FIG. 8. *Evolution of the total communicability when 50 downdates, updates or rewires are performed on two synthetic networks with $n = 1000$ nodes.*

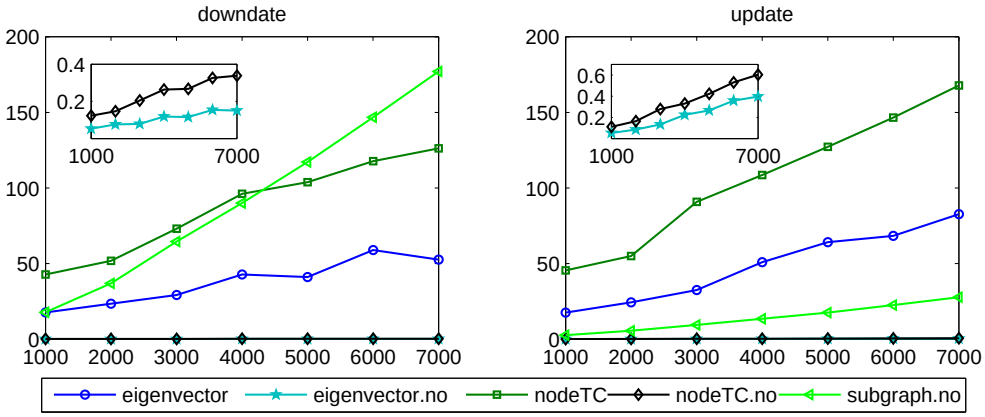


to nodes already in the network with a probability proportional to the degree of the already existing nodes. This results in a scale–free degree distribution. Note that with this construction, the minimum degree of the network is d. When $d > 1$ this means that the network has no dangling nodes (nodes of degree 1), whereas in many real-world networks one often observes a high number of dangling nodes.

The second class of synthetic test matrices used in our experiments corresponds to Watts–Strogatz small world networks. The small world model was developed as a way to impose a high clustering coefficient onto classical random graphs [41]. The function used to build these matrices takes the form `smallw(n,k,p)`. Here n is the number of nodes in the network, originally arranged in a ring and connected to their k nearest neighbors. Then each node is considered independently and, with probability p, an edge is added between the node and one of the other nodes in the graph, chosen uniformly at random (self-loops and multiple edges are not allowed). In our tests, we have used matrices with $n = 1000$ nodes which were built using the default values for the functions previously described. More in detail, we used $d = 2$ in the Barabási–Albert model and $k = 2$, $p = 0.1$ in the Watts–Strogatz model.

The results for our tests are presented in Figure 8. These results agree with what we have seen previously on real-world networks. Interestingly, `degree` does not perform well for the downdate when working on the preferential attachment model; this

FIG. 9. *Timings for scale-free graphs of increasing size (500 modifications).*



behavior reflects what we have seen for the networks USAir97, as–735, and Erdös02, which are indeed scale–free networks.

5.3. Timings for synthetic networks. We have performed some experiments with synthetic networks of increasing size in order to assess the scalability of the various methods introduced in this paper. A sequence of seven adjacency matrices corresponding to Barabási–Albert scale-free graphs was generated using the CONTEST toolbox. The order of the matrices ranges from 1000 to 7000; the average degree is kept constant at 5. A fixed number of modifications ($K = 500$) was carried out on each network. All experiments were performed using Matlab Version 7.12.0.635 (R2011a) on an IBM ThinkPad running Ubuntu 12.04.5 LTS, a 2.5 GHZ Intel Core i5 processor, and 3.7 GiB of RAM. We used the built-in Matlab function `eigs` (with the default settings) to approximate the dominant eigenvector of the adjacency matrix A, the Matlab toolbox `mmq` [31] to estimate the diagonal entries of e^A (with a fixed number of five nodes in the Gauss–Radau quadrature rule, hence five Lanczos steps per estimate), and the toolbox `funm_kryl` to compute the vector $e^A\mathbf{1}$ of total communicabilities, also with the default parameter settings.

The results are shown in Figure 9. The approximate (asymptotic) linear scaling behavior of the various methods (in particular of `nodeTC.no` and `eigenvector.no`, which are by far the fastest, see the insets) is clearly displayed in these plots.

5.4. Timings for larger networks. In Tables 2–3 we report the timings for various methods when $K = 2000$ downdates and updates are selected for the three largest networks listed in Table 1.

The timings presented refer to the selection of the edges to be downdated or updated, which dominates the computational effort. For the method `subgraph.no` in the case of downdates, we restricted the search of candidate edges to a subset of E in order to reduce costs. For the three test networks we used 40%, 45% and 15% of the nodes, respectively, chosen by taking those with lowest eigenvector centrality, and the corresponding edges. We found the results to be very close to those obtained working with the complete set E, but at a significantly lower cost (especially for the largest network).

These results clearly show that algorithms `nodeTC.no` and `eigenvector.no` are

TABLE 2
Timings in seconds for $K = 2000$ downdates performed on the three largest networks in Table 1.

	ca–HepTh	as–22july06	usroad–48
`eigenvector`	278.13	599.83	11207.39
`eigenvector.no`	0.07	1.79	4.08
`nodeTC`	553.04	1234.49	2634.27
`nodeTC.no`	0.34	0.83	1.34
`subgraph.no`	107.36	383.34	1774.07
`degree`	29.67	53.42	153.52

TABLE 3
Timings in seconds for $K = 2000$ updates performed on the three largest networks in Table 1

	ca–HepTh	as–22july06	usroad–48
`eigenvector`	192.8	436.9	1599.5
`eigenvector.no`	0.19	0.33	5.85
`nodeTC`	561.9	1218.8	2932.
`nodeTC.no`	0.30	0.55	1.59
`subgraph.no`	3.13	7.20	121.4
`degree`	11.1	12.4	175.8

orders of magnitude faster than the other methods; method `subgraph.no`, while significantly more expensive, is still reasonably efficient[3] and can be expected to give better results in some cases (e.g., on networks with a very small spectral gap). The `degree` algorithm, on the other hand, cannot be recommended in general since it gives somewhat inferior results. The remaining methods `eigenvector`, `nodeTC` and `subgraph` (not shown here) are prohibitively expensive for large networks, at least when the number K of modifications is high (as it is here).

We also observe that downdating is generally a more expensive process than updating, since in the latter case the edges are to be chosen among a fairly small subset of all virtual edges, whereas in the downdating process we work on the whole set E of existing edges (or on a large subset of E). For some methods the difference in cost becomes significant when the networks are sufficiently large and the number of modifications to be performed is high.

Summarizing, the method labelled `nodeTC.no` is the fastest and gives excellent results, quite close to those of the more expensive methods, and therefore we can recommend its use for the type of problems considered here. The methods labelled `eigenvector.no` and `subgraph.no` are also effective and may prove useful in some settings, especially for updating.

6. Evolution of other connectivity measures. In this section we want to highlight another facet of the methods we have introduced for (approximately) optimizing the total communicability. In particular, we look at the evolution of other network properties under our updating strategies. When building or modifying a network, there are various features that one may want to achieve. Typically, there are two main desirable properties: first, the network should do a good job at spreading information, i.e., have a high total communicability; second, the network should be robust under targeted attacks or random failure, which is equivalent to the requirement that it should be difficult to "isolate" parts of the network, i.e., the network should be "well connected". This latter property can be measured by means of var-

[3]It is worth mentioning that in principle it is possible to greatly reduce the cost of this method using parallel processing, since each subgraph centrality can be computed independently of the others.

ious indices. One such measure is the spectral gap $\lambda_1 - \lambda_2$. As a consequence of the Perron–Frobenius Theorem, adding an edge to a connected network causes the dominant eigenvalue λ_1 of A to increase. Test results (not shown here) show that when a network is updated using one of our techniques, the first eigenvalue increases rapidly with the number of updates. On the other hand, the second eigenvalue λ_2 tends to change little with each update and it may even decrease (recall that the matrix $UW^T = \mathbf{e}_i\mathbf{e}_j^T + \mathbf{e}_j\mathbf{e}_i^T$ being added to A in an update is indefinite). Therefore, the spectral gap $\lambda_1 - \lambda_2$ widens rapidly with the number of updates.[4] It has been pointed out by some authors (see, e.g., [15, 37]) that a large spectral gap is typical of complex networks with good expansion properties.

Here we focus on a related measure, the natural connectivity. In particular, we investigate the effect of our proposed methods of network updating on the evolution of this index.

6.1. The natural connectivity. In [28] the authors introduced a measure of network connectivity which is based on an intuitive notion of robustness and whose analytical expression has a clear meaning and can be derived from the eigenvalues of A; see also [42]. The idea underlying this index is that a network is more robust if there exists more than one route to get from one node to another; this property ensures that if a route become unusable, there is an alternative way to get from the source of information to the target. Therefore, intuitively a network is more robust if it has a lot of (apparently) redundant routes connecting its vertices or, equivalently, if each of its nodes is involved in a lot of closed walks. The natural connectivity aims to measure this property by taking advantage of the fact that a measure for the total number of closed walks in a network already exists: the Estrada index [14]. Recall that the *Estrada index* of a graph is defined as

$$EE(G) = \sum_{i=1}^{n} e^{\lambda_i}.$$

Normalizing this value and taking the natural logarithm, one obtains the *natural connectivity* (or *natural eigenvalue*) of the graph,

$$\overline{\lambda}(A) = \ln\left(\frac{EE(G)}{n}\right),$$

which can be seen as an "average" eigenvalue and changes monotonically when an edge is downdated or updated in the graph (see [28]). Coarse bounds on this index are readily obtained:

$$0 \le \overline{\lambda}(A) \le \ln((n-1)e^{-1} + e^{n-1}) - \ln n.$$

The lower bound is attained by the empty graph, while the upper bound is attained by the complete graph, as a straightforward computation shows. Using the results in [4] we obtain more refined bounds via quadrature rules:

$$\ln\left(\frac{1}{n}\sum_{i=1}^{n}\frac{\beta^2 e^{\frac{d_i}{\beta}} + d_i e^{-\beta}}{\beta^2 + d_i}\right) \le \overline{\lambda}(A) \le \ln\left(\frac{1}{n}\sum_{i=1}^{n}\frac{\alpha^2 e^{\frac{d_i}{\alpha}} + d_i e^{-\alpha}}{\alpha^2 + d_i}\right),$$

[4] This fact, incidentally, may serve as further justification for the effectiveness of algorithms like `nodeTC.no` and `eigenvector.no`.

Algorithm 1: Updating algorithm from [11].

Data: A adjacency matrix and $K \in \mathbb{N}$
Result: Set S of K edges to be added
$S = \emptyset$;
Compute the top t eigenpairs $(\lambda_k, \mathbf{q}_k)$ of A;
for *iter = 1 : K* **do**
  Compute $d_{\max} = \max(d_i)$, the largest degree of A ;
  Find the set C of $d_{\max}$ nodes with the highest eigenvector centrality;
  Select the edge $(i^*, j^*) \in \overline{E}$ that maximizes

$$e^{\lambda_1}\left(e^{2\mathbf{q}_1(i)\mathbf{q}_1(j)} + \sum_{h=2}^{t} e^{\lambda_h - \lambda_1} e^{2\mathbf{q}_h(i)\mathbf{q}_h(j)}\right)$$

  and such that $i^*, j^* \in C$, $i^* \neq j^*$;
  $S = S \cup \{(i^*, j^*)\}$, $E = E \cup \{(i^*, j^*)\}$;
  Update A;
  Update the top t eigenpairs as

$$\begin{cases} \lambda_k = \lambda_k + 2\mathbf{q}_k(i)\mathbf{q}_k(j); \\ \mathbf{q}_k = \mathbf{q}_k + \sum_{h \neq k}\left(\frac{\mathbf{q}_h(i)\mathbf{q}_k(j) - \mathbf{q}_k(i)\mathbf{q}_h(j)}{\lambda_k - \lambda_h}\mathbf{q}_h\right) \end{cases} \qquad k = 1, 2, \ldots, t;$$

end
Return S.

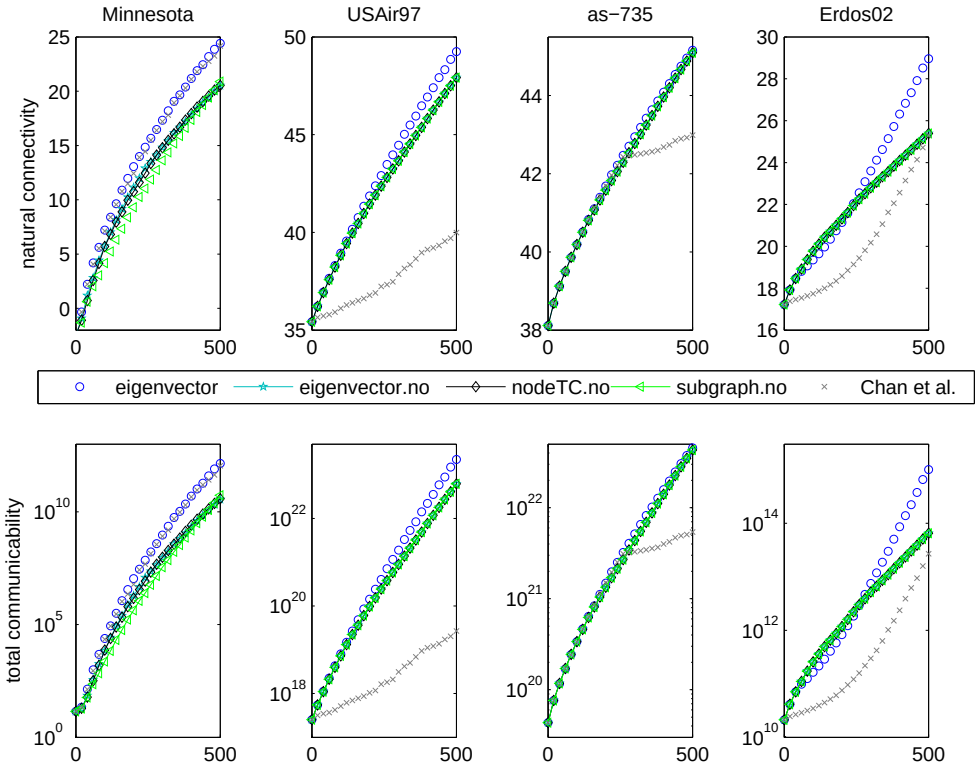
where $[\alpha, \beta]$ is an interval containing the spectrum of $-A$ and d_i is the degree of node i. This index has been recently used (see [11]) to derive manipulation algorithms that directly optimize this robustness measure. In particular, the updating algorithm introduced in [11] appears to be superior to existing heuristics, such as those proposed in [9, 21, 38]. This algorithm, which costs $O(mt + Kd_{max}^2 t + Knt^2)$ where $d_{max} = \max_{i \in V} d_i$ and t is the (user-defined) number of leading eigenpairs, selects K edges to be added to the network as described in Algorithm 1.

We have compared our updating techniques with that described in Alg. 1. Results for four representative networks are shown in Figure 10. In our tests, we used the value $t = 50$ (as in [11]), and we select $K = 500$ edges. Note that, when K is large, the authors recommend to recompute the set of t leading eigenpairs every l iterations. This operation requires an additional effort that our faster methods do not need. Since the authors in [11] show numerical experiments in which the methods with and without the recomputation return almost exactly the same results, we did not recompute the eigenpairs after any of the updates.

Figure 10 displays the results for both the evolution of the natural connectivity and of the normalized total communicability, where the latter is plotted in a semi–logarithmic scale. A total of 500 updates have been performed. The method labelled `Chan` selects the edges according to Algorithm 1 choosing from all the virtual edges of the graph. For our methods we used, as before, the virtual edges in the subgraph obtained selecting the top 10% or 20% of nodes ranked according to the eigenvector centrality. As one can easily see, our methods generally outperform the algorithm proposed in [11]. In particular, `nodeTC.no` and `eigenvector.no` give generally better results than `Chan` and are much faster in practice. For instance, the execution time with `Chan` on the network ca-HepTh was over 531 seconds, and much higher for the two larger networks. We recall (see Table 3) that the execution times for `nodeTC.no` and `eigenvector.no` are are about three orders of magnitude smaller.

It is striking to see how closely the evolution of the natural connectivity mirrors

FIG. 10. *Evolution of the natural connectivity and of the normalized total communicability (in a semi–logarithmic scale plot) when up to 500 updates are performed on four real-world networks.*



the behavior of the normalized total communicability. This is likely due to the fact that both indices depend on the eigenvalues of A (with a large contribution coming from the terms containing λ_1, see (2.1)), and all the updating strategies used here tend to make λ_1 appreciably larger.

These findings indicate that the (normalized) total communicability is equally effective an index as the natural connectivity for the purpose of characterizing network connectivity. Since the network total communicability can be computed very fast (in $O(n)$ time), we believe that the normalized total communicability should be used instead of the natural connectivity, especially for large networks. Indeed, computing the natural connectivity requires evaluating all the diagonal entries of e^A and is therefore significantly more expensive, for large networks, than the total communicability.

7. Conclusions and future work. In this paper we have introduced several algorithms that can be used to construct undirected networks that are highly sparse, and yet have a large total communicability. These same algorithms can also be used to modify an existing network in order to increment as much as possible the total network communicability when adding or rewiring an edge, and to decrease this index as little as possible when removing an edge, to a very good approximation.

The algorithms are based on different heuristics and use measures of edge centrality, several of which have been introduced in this work. The heuristics, far from being *ad hoc*, are widely applicable and mathematically justified. All our algorithms can be

implemented using well-established tools from numerical linear algebra: algorithms for eigenvector computation, Gauss-based quadrature rules for estimating quadratic forms, and Krylov subspace methods for computing the action of a matrix function on a vector. At bottom, the Lanczos algorithm is the main player. High quality, public domain software exists to perform these modifications efficiently.

Among all the methods introduced here, the best results are obtained by the `nodeTC.no` and `eigenvector.no` algorithms, which are based on the edge total communicability and eigenvector centrality, respectively. These methods are extremely fast and returned excellent results in virtually all the tests we have performed. For updating networks characterized by a small spectral gap, a viable alternative is the algorithm `subgraph.no`. While more expensive than `nodeTC.no` and `eigenvector.no`, this method scales linearly with the number of nodes and yields consistently good results.

Finally, we have shown that the total communicability can be effectively used as a measure of network connectivity, which plays an important role in designing robust networks. Indeed, the total communicability does a very good job at quantifying two related properties of networks: the ease of spreading information, and the extent to which the network is "well connected". Our results show that the total communicability behaves in a manner very similar to the natural connectivity under network modifications, while it can be computed much more quickly.

Future work should include the extension of these techniques to other types of networks, including directed and weighted ones.

Acknowledgements. We are grateful to Ernesto Estrada for providing some of the networks used in the numerical experiments and for pointing out some useful references. The first author would like to thank Emory University for the hospitality offered in 2014, when this work was completed.

REFERENCES

[1] M. Afanasjew, M. Eiermann, O. G. Ernst, and S. Güttel, *Implementation of a restarted Krylov subspace method for the evaluation of matrix functions*, Linear Algebra Appl., 429 (2008), pp. 2293–2314.
[2] A.-L. Barabási, *Linked: The New Science of Networks*, Perseus, Cambridge, 2002.
[3] A. L. Barabási and R. Albert, *Emergence of scaling in random networks*, Science, 286 (1999), pp. 509–512
[4] M. Benzi and P. Boito *Quadrature rule-based bounds for functions of adjacency matrices*, Linear Algebra Appl. 433 (2010), pp. 637–652.
[5] M. Benzi and G. H. Golub *Bounds for the entries of matrix functions with application to preconditioning*, BIT 39 (1999), pp. 417–438.
[6] M. Benzi and C. Klymko, *Total communicability as a centrality measure*, J. Complex Networks, 1(2) (2013), pp. 124–149.
[7] M. Benzi and C. Klymko, *A matrix analysis of different centrality measures*, arXiv:1312.6722v3, 2014.
[8] M. W. Berry, T. P. Chartier, K. R. Hutson, and A. N. Langville, *Identifying influential edges in a directed network: big events, upsets and non-transitivity*, J. Complex Networks, 2(2) (2013), pp. 87–109.
[9] A. Beygelzimer, G. Grinstein, R. Linsker, and I. Rish, *Improving network robustness by edge modification*, Physica A, 357 (2005), pp. 593–612.
[10] U. Brandes and T. Erlebach, eds., *Network Analysis: Methodological Foundations*, Lecture Notes in Computer Science Vol. 3418, Springer, New York, 2005.
[11] H. Chan, L. Akoglu, and H. Tong, *Make it or break it: manipulating robustness in large networks*, Proceedings of the 2014 SIAM Data Mining Conference (2014), Society for Industrial and Applied Mathematics, pp. 325–333.

[12] R. COHEN AND S. HAVLIN, *Complex Networks: Structure, Robustness and Function*, Cambridge University Press, Cambridge, UK, 2010.
[13] T. DAVIS AND Y. HU, *The University of Florida Sparse Matrix Collection*, http://www.cise.ufl.edu/research/sparse/matrices/.
[14] E. ESTRADA, *Characterization of 3D molecular structure*, Chem. Phys. Lett. (319): 713 (2000).
[15] E. ESTRADA, *Spectral scaling and good expansion properties in complex networks*, Europhys. Lett., 73 (2006), pp. 649–655.
[16] E. ESTRADA, *The Structure of Complex Networks. Theory and Applications*, Oxford University Press, 2012.
[17] E. ESTRADA AND N. HATANO, *Communicability in complex networks*, Phys. Rev. E, 77 (2008), 036111.
[18] E. ESTRADA, N. HATANO, AND M. BENZI, *The physics of communicability in complex networks*, Phys. Rep., 514 (2012), pp. 89–119.
[19] E. ESTRADA AND J. A. RODRÍGUEZ-VELÁZQUEZ, *Subgraph centrality in complex networks*, Phys. Rev. E 71 (2005), 056103.
[20] C. FENU, D. MARTIN, L. REICHEL, AND G. RODRIGUEZ, *Network analysis via partial spectral factorization and Gauss quadrature*, SIAM J. Sci. Comput, 35 (4) (2012), pp. A2046–A2068.
[21] H. FRANK AND I. FRISCH, *Analysis and design of survivable networks*, IEEE Trans. Comm. Tech., 18 (5) (1970), pp. 501–519.
[22] G. H. GOLUB AND G. MEURANT, *Matrices, Moments and Quadrature with Applications*, Princeton University Press, Princeton, NJ 2010.
[23] S. GÜTTEL, `funm_kryl` *toolbox for Matlab*, www.mathe.tu-freiberg.de/∼ guttels/funm_kryl/.
[24] P. E. HART, N. J. NILSSON, AND B. RAPHAEL, *A formal basis for the heuristic determination of minimum cost paths*, IEEE Trans. Systems Sci. Cybernetics, 4 (2) (1968), pp. 100–107.
[25] N. J. HIGHAM, *Function of Matrices, Theory and Computation*, Society for Industrial and Applied Mathematics, Philadelphia, PA, 2008.
[26] S. HOORY, N. LINIAL, AND A. WIGDERSON, *Expander graphs and their applications*, Bull. Amer. Math. Soc., 43 (2006), pp. 439–561.
[27] R. A. HORN AND C. R. JOHNSON, *Matrix Analysis. Second Edition*, Cambridge University Press, 2013.
[28] W. JUN, M. BARAHONA, Y. TAN, AND H. DENG, *Natural connectivity of complex networks*, Chinese Physical Letters, 27 (2010), 078902.
[29] V. H. P. LOUZADA, F. DAOLIO, H. J. HERRMANN, AND M. TOMASSINI, *Smart rewiring for network robustness*, J. Complex Networks, 1(2) (2013), pp. 150–159.
[30] D. LUSSEAU, K. SCHNEIDER, O. J. BOISSEAU, P. HAASE, E. SLOOTEN, AND S. M. DAWSON, *The bottlenose dolphin community in Doubtful Sound features a large proportion of long-lasting associations*, Behavioral Ecology and Sociobiology 54 (2003), pp. 396–405.
[31] G. MEURANT, *MMQ toolbox for MATLAB*, http://pagesperso-orange.fr/gerard.meurant/.
[32] C. D. MEYER, *Matrix Analysis and Applied Linear Algebra*, SIAM, Philadelphia, 2000.
[33] J. H. MICHAEL AND J. G. MASSEY, *Modeling the communication network in a sawmill*, Forest Products Journal, 47 (1997), pp. 25–30.
[34] M. E. J. NEWMAN, *Networks. An Introduction*, Oxford University Press, 2010.
[35] W. DE NOOY, A. MRVAR, AND V. BATAGELJ, *Exploratory Social Network Analysis with Pajek*, Cambridge University Press, 2004.
[36] A. PINAR, J. MEZA, V. DONDE, AND B. LESIEUTRE, *Optimization strategies for the vulnerability analysis of the electric power grid*, SIAM J. Optimiz., 20 (4) (2010), pp. 1786–1810.
[37] D. PUDER, *Expansion of Random Graphs: New Proofs, New Results*, arXiv:1212.5216v2, 2013.
[38] B. SHARGEL, H. SAYAMA, I. R. EPSTEIN, AND Y. BAR-YAM, *Optimization of robustness and connectivity in complex networks*, Phys. Rev. Letters, 90 (6) (2003), 068701.
[39] A. TAYLOR AND D. J. HIGHAM, *CONTEST: Toolbox files and documentation*, http://www.mathstat.strath.ac.uk/research/groups/ numerical_analysis/contest/toolbox.
[40] A. TAYLOR AND D. J. HIGHAM, *CONTEST: A controllable test matrix toolbox for Matlab*, ACM Trans. Math. Softw., 35 (2009), pp. 26:1–26:17.
[41] D. J. WATTS AND S. H. STROGATZ, *Collective dynamics of 'small-world' networks*, Nature, 393 (1998), pp. 440–442.
[42] J. WU, M. BARAHONA, Y. TAN, AND H. DENG, *Robustness of random graphs based on graph spectra*, Chaos, 22 (2012), 043101.
[43] W. W. ZACHARY, *An information flow model for conflict and fission in small groups*, J. Anthropol. Res., 33 (1977), pp. 452–473.
[44] L. D. ZELENY, *Adaptation of research findings in social leadership to college classroom procedures*, Sociometry, 13 (4) (1950), pp. 314–328